Documentation

Project 2
EIT060 Computer Security
Grupp 22

dat13tpo Tom Postema
dat13abu Anders Buhl
dic14jan Joakim Andersson
dic14hpe Henrik Persson

February 25, 2016

1 High-level architectural overview

An overview of the system is given by figure 1.

1.1 Client

The client uses a certificate signed by a CA and generated by an administrator. Every client will use a different keystore containing their own certificate and the CA certificate. Every client will use the same truststore containing the CA to verify the server. To be able to use the certificate the user has to know the password for the certificate.

The client uses an ID and a password to access a certificate. The ID is used to choose which keystore to use and the password is the password for the keystore and the certificate.

Each certificate contains information about the user: ID, name, division and title. This will be used by the server when logging in for the first time to create a new user.

The client verifys the server certificate and connection is established. The client tries to login. When logged in the client can send commands to the server to perform wanted operations. The server then performs the requested operation or responds with the requested information or a denial message.

1.2 Server

The server is authenticated with its own certificate signed by the CA. The server keystore contains the CA and the server certificate. The server truststore contains the CA. When a client tries to connect to the server, the server demands the client to send a certificate. The client's certificate is verified and connection is established. The client then tries to login. If the certificate contains errors the client is denied access.

When the client successfully logged in the server accepts commands. The server receives the commands and sends them to the Hospital class which checks if the user is allowed to perform the requested operation or access the requested information. If the operation is to alter, read, create or delete a record the hospital class checks if the user is allowed to perform the operation by observing what type of division and records the user is associated to.

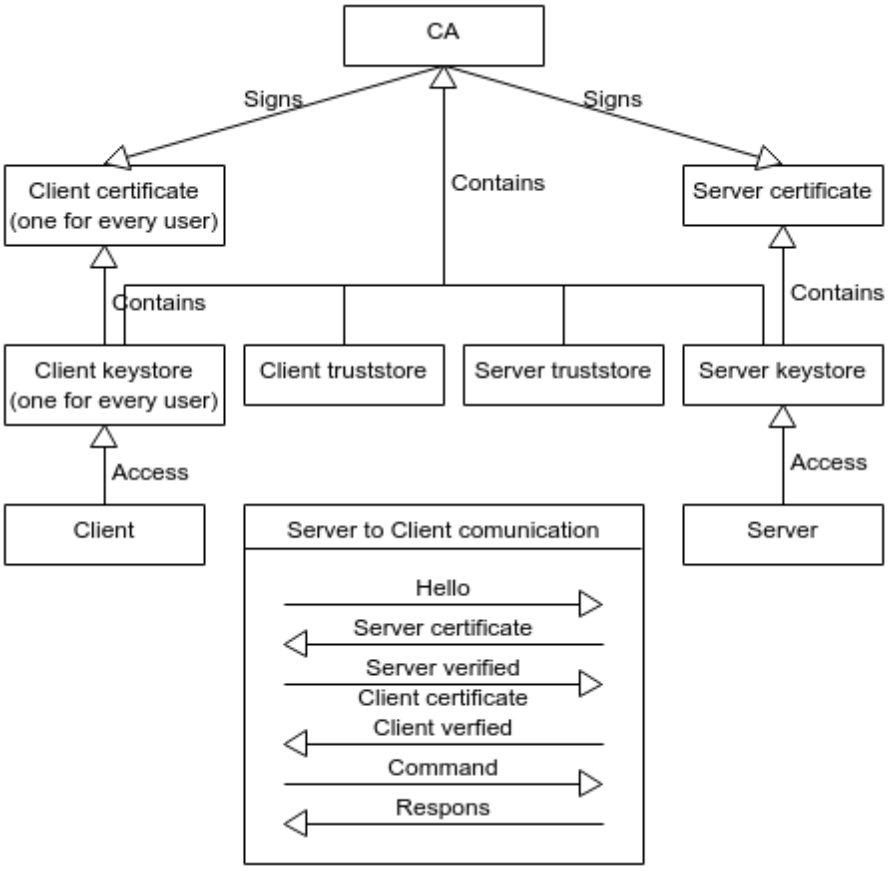

Figure 1: Overview of the design of the system.

2 Ethical discussion

Clients in the system are patients, doctors, nurses and a government agent. In the implemented access control scheme, access is given when the following rules are obliged:

- A patient is allowed to read his/her own list of records.
- A nurse may read and write to all records associated with him/her, and also read all records associated with the same division.
- A doctor may read and write to all records associated with him/her, and also read all records associated with the same division. In addition, the doctor can create new records for a patient provided that the doctor is treating the patient. When creating the record, the doctor also associates a nurse with the record.
- A government agency is allowed to read and delete all types of records.

An access control scheme that provides the best trade-off between confidentiality, integrity and availability is a context-based control scheme. Access is only allowed during office hours and from the division associated with the record. It is assumed that the hospital staff can only access workspaces at the division where they work. Furthermore, the following rules must be obliged:

- A patient is allowed to read his/her own list of records.
- A nurse may read all records associated with the division they work at.
- A doctor may write to all records associated with him/her and read all records associated with the division they work at. A doctor treating a patient can create new records. The doctor then also associates a nurse with the record.
- A government agency is allowed to delete all types of records.

Compared to the implemented access control scheme, a nurse can no longer write to a record and a government agency is only allowed to delete records in order to increase confidentiality. To increase integrity, it is now impossible to get access any time of the day and you are only allowed access from the correct workspace.

The priorities of the hospital administration are probably that the system is economical and available.

As an engineer or security expert, making sure that the system doesn't have any vulnerabilities is a priority. This desire might be limited by the budget the hospital administration has put up.

As the system end-user, privacy and patient-health are important aspects of the system. Once again the hospital administration's budget might limit how private the system is.

3 Security evaluation of the design

3.1 The system

The system uses certificates to verify both the server and the client. Each client uses its own certificate. Theses certificates are distributed by the administrator and can be transferred via USB or similar methods. The keystore and certificate belonging to the client is protected by a password. This is the two factors of the authentication. The first factor is the certificate which the user needs to own and the second is the password which the user needs to know.

3.1.1 Cipher suite

The cipher suite chosen when running the program is:
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

The different parts of the cipher suite are[3]:

- TLS - Protocol used
- DHE - Diffie-Hellman Ephemeral
  Diffie-Hellman Ephemeral is the key agreement protocol used by the system. No pre-shared secret is needed instead large primes are used to determine a key. The values are signed with the private key[1].
- DSS - Digital Signature Standard
  Digital Signature Standard is the authentication algorithm used by the system.
- AES_256_CBC - Symmetric Encryption Algorithm (256 bits)
- SHA256 - Hashing algorithm
  Secure Hash Algorithm is the hashing algorithm used by the system.

3.2 Attacks

- Brute Force
  Almost every system can be somewhat broken by brute force. However, the keytool insists on the passwords to contain at a minimum 6 characters in order to work, which we increased to 8 characters minimum for our program. This is combined with requiring at least one uppercase letter, one lowercase letter, one digit and a special symbol (out of the 32 available by ASCII source [2]). This makes the least amount of possible combinations for a password 95^8. We could not

however implement a fitting lockout system to prevent a brute force-attack from using the terminal however, so password guessing might also be an issue. However, if the password were to be obtained the certificate is still needed.

- Dictionary Attack
  Since we allow our users to select their password by themselves, we can't effectively prevent them from using dictionary words as passwords, other than how we set the requirement for the passwords to include an uppercase letter, a lowercase letter, a digit and a special symbol. They may still use dictionary words but we have increased the amount of variation the attackers have to include in their script for it to match existing words, as they often account for variations in passwords like 'p4ssw0rd' or similar. And after they got the password they still need the user's certificate.

- Time-Memory Tradeoff-attack
  Since the keystore stores passwords using salts, using time-memory tradeoff loses its effectiveness as it can only use their table for one salt at a time. And after they got the password they still need your certificate. [5]

- Buffer Overflow
  As the program is written in Java it is not susceptible to buffer overflows. However, since the Java Virtual Machine we use is programmed partly in C it might be at risk for attacks. This we have not implemented a defence against. [6]

- Social Engineering
  As is the case in most systems, the biggest flaw might be human error. As we cannot implement anything against a user using a post-it note on the screen with their login information (Users might be especially prone to do so since we have stricter password requirements with a mandatory digit, a lower and uppercase letter and a special symbol). We have chosen to include a text on the login screen which warns the user from ever letting someone else know their password. As a big risk is someone seemingly authorized to ask for the users password might obtain it from the user. If the attacker succeeds with any of this, they then have your password, but they still need to access the system through your terminal or steal your certificate through a flashdrive or similar something which they might be able to get, posing as a member of the technical support staff.

- Spoofing-Attacks
  If an attacker manages to create a program that looks identical or at least similar to our client, and install it on the computer which the

user has his certificate on, then make the user use his program to get his login credentials. Then as for the rest of the attacks, he still needs to use the same computer to use the victims certificate and/or steal the certificate file. We could not efficiently implement a way for the user to check how many failed login attempts the user has had before, so the user will not know if they have been victim to a spoofing attack.

- Theft
  There can be break-ins in attempt to steal the certificates. A stolen certificate can then be used to try to break the password using one of the password-breaking methods discussed above. We could not implement an efficient way to change the certificates and ban the stolen certificates.

References

[1] Hell, Martin. "Key establisment". Lecture slides on key establisment [2016-02-25]

[2] KnowledgeBase. Last modified 28 July 2016. "Learn@UW - Use of Special Characters in Passwords". `https://kb.wisc.edu/page.php?id=4073` [2016-02-23]

[3] The Sprawl. Published September 20, 2009. "TLS and SSL cipher suites" `http://www.thesprawl.org/research/tls-and-ssl-cipher-suites/` [2016-02-23]

[4] Wikipedia. Last modified 5 February 2016. "Sample X.509 certificates". `https://en.wikipedia.org/wiki/X.509\#Sample_X.509_certificates` [2016-02-23]

[5] OpenJDK - sun.security.provider.KeyProtector documentation `http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/6-b27/sun/security/provider/KeyProtector.java` [2016-02-25]

[6] Wikipedia. Last modified 19 February 2016. "OpenJDK" `https://en.wikipedia.org/wiki/OpenJDK` [2016-02-25]